UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/676,227 | 09/30/2003 | Gunter Schwarzbauer | 06896/100M097-US1 | 9707 |

7278 7590 06/06/2008

DARBY & DARBY P.C.
P.O. BOX 770
Church Street Station
New York, NY 10008-0770

| EXAMINER | |
|---|---|
| RUTLEDGE, AMELIA L | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2176 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 06/06/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 10/676,227
Filing Date: September 30, 2003
Appellant(s): SCHWARZBAUER ET AL.

Kevin J. Beach
For Appellant

**EXAMINER'S ANSWER**

This is in response to the amended appeal brief filed 03/24/2008 appealing from the Office action mailed 07/14/2006.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

| | | |
|---|---|---|
| 6,549,944 B1 | WEINBERG ET AL. | 4-2003 |
| US PUB. NO. 2002/0062342 | SIDLES | 5-2002 |

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

***Claim Rejections - 35 USC § 102***

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

**Claims 1-12 and 29 are rejected under 35 U.S.C. 102(e) as being anticipated by Weinberg et al. ("Weinberg"), U.S. Patent No. 6,549,944 B1, issued April 2003.**

**Regarding independent claim1,** Weinberg teaches an automated testing tool for testing and monitoring web applications (Col. 2, l. 30 – Col. 3, l. 37; Col. 8, l. 40-67); compare to *A system for automatic context management for testing, monitoring and*

*automating a network application having client side executable code.* Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33); compare to *a tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user.* Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37). Weinberg teaches that the recorder and replay engine have a context full API, since the test scenario is generated using information specific to the site context, i.e., server access log files and associated site usage information (Col. 33, l. 40-Col. 35, l. 20).

**Regarding dependent claim 2,** Weinberg teaches a page-level API (Col. 19, l. 11-50; Fig. 8).

**Regarding dependent claims 3-6,** Weinberg teaches an extensible document parser operable to determine at least one parser extension, where the parser extension includes a replay instruction specifying at least one parser addition and parameters for the parser addition, since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60), including the Action Tracker and Load Wizard containing replay instructions specifying parser additions; as well as parser extensions in the form of other external client applications (Col. 18, l. 40-59).

**Regarding dependent claims 7-10**, Weinberg teaches a library of parser additions, since Weinberg teaches the addition multiple plug-ins, each of which implements a specific parsing algorithm (Col. 17, l. 53-Col. 18, l. 20), including an algorithm for parsing hyperlinks, the Link Doctor. Weinberg teaches an algorithm for parsing HTML forms, the Dynamic Scan feature (Col. 25, l. 50-Col. 26, l. 51), including different configurations, i.e., extensions, of the Dynamic Scan feature including a standard web browser or a special integrated web browser (Col. 26, l. 41-51).

**Regarding dependent claim 11,** Weinberg teaches a parser addition which implements an algorithm for parsing embedded documents from an HTML document, since Weinberg teaches a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33); and the forms were embedded in HTML documents.

**Regarding dependent claim 12,** Weinberg teaches a method of parsing hyperlinks by searching text between a left and right boundary string (Col. 20, l. 62-Col. 21, l. 60); it is inherent in the teaching of Weinberg that this method would have been used by the plug-ins, since Weinberg teaches an API with objects including information about the URLs and links (Col. 19, l. 11-25).

**Regarding independent claim 29**, Weinberg teaches an automated testing tool for testing and monitoring web applications (Col. 2, l. 30 – Col. 3, l. 37; Col. 8, l. 40-67); including a processor and memory (Col. 7, l. 51-Col. 8, l. 12; claim 13). Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML

code, i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33); compare to *said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user.* Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37). Weinberg teaches that the recorder and replay engine have a context full API, since the test scenario is generated using information specific to the site context, i.e., server access log files and associated site usage information (Col. 33, l. 40-Col. 35, l. 20).

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

**Claims 13-28, 30, and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Weinberg, as applied to claims 1-12 and 29 above, in view of Sidles, U.S. Pub. No. 2002/0062342, published May 2002.**

**Regarding dependent claims 13 and 14**, Weinberg teaches that the API comprises form replay instructions, since Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable

code, which may be set to automatically fill in dynamic forms without interaction from a user, when replayed (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33). Weinberg does not explicitly teach form merging instructions including instructions for merging an HTML form and one context-full script form to produce a form to be submitted. However, Sidles teaches a reference to a previously downloaded form in a test script, i.e., to see if the user made changes to a field, and instructions for merging the form and scripts containing previously submitted form information to automatically fill out a form to be submitted (p. 12, par 120-123). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 15**, while Weinberg does not explicitly teach instructions for merging each individual form field, Sidles teaches merging instructions for each individual form field of the HTML form and script form (p. 10-11, par. 102-104). Sidles teaches instructions for matching each field of the form to a dictionary entry for the form, and if a match cannot be found, the field information is not sent, compare to *an instruction to send a form field value obtained from said HTML form; an instruction to send a form field value specified in said script form; and an instruction to not send one of said form fields*. Both Weinberg and Sidles are analogous art, being directed toward

the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 16**, Weinberg teaches associating a database query with a URL of a site and automatically performing the query and mapping the results next time an automatic update is performed (Col. 24, l. 1-19); compare to *an action URL in said test script instead of an action URL obtained from said HTML form for said form to be submitted*.

**Regarding dependent claim 17**, Weinberg recording a page level test script comprising parser extensions, since Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37), and since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60). Weinberg does not explicitly teach form merging instructions including instructions for merging an HTML form and one context-full script form to produce a form to be submitted. However, Sidles teaches a reference to a previously downloaded form in a test script, i.e., to see if the user made changes to a field, and instructions for merging the form and scripts containing previously submitted form information to automatically fill out a form to be submitted (p. 12, par 120-123). Both Weinberg and Sidles are

analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 18**, Weinberg teaches that testing tool has a recorder to record session history by building representations of all web pages downloaded so far during a recording session, because the Dynamic Scan feature records session history and builds representations of all web pages downloaded (Col. 23, l. 15-Col. 26, l. 19). Weinberg also teaches a map representation of recorded pages from a site session.

**Regarding dependent claims 19 and 20**, Weinberg teaches a page-level API (Col. 19, l. 11-50; Fig. 8). Weinberg teaches an extensible document parser operable to determine at least one parser extension, where the parser extension includes a replay instruction specifying at least one parser addition and parameters for the parser addition, since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60), including the Action Tracker and Load Wizard containing replay instructions specifying parser additions; as well as parser extensions in the form of other external client applications (Col. 18, l. 40-59). Weinberg teaches that the recorder utilizes the document parsers. While Weinberg does not explicitly teach that the recorder automatically detects which form merging instructions are needed in order to

record a test script which will correctly use dynamic information during a script replay, Sidles teaches an automatic method of form filling where the system tests which rules and instructions are needed in order to correctly fill a dynamic form (p. 12, par 125). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 21**, while Weinberg does not explicitly teach that the recorder detects the need for recording at least one of said parser extensions by detecting that a context-less replay instruction is to be recorded otherwise, Sidles teaches a process performed by the completed form analysis engine for detecting the need for recording parser extensions, i.e., dictionary or database entries (p. 11, par. 107-113) and detecting whether instructions should be recorded with or without context (p. 11, par. 110-112). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an

self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 22**, claim 22 is directed toward substantially similar subject matter as claimed in dependent claims 3, 4, 7, and 8 and is rejected along the same rationale, except where Claim 22 recites the additional limitation*: said recorder is operable to detect which one of said parser extensions is to be recorded by querying each of said parser additions for suitable parameters.* While Weinberg does not explicitly teach the claimed limitation, Sidles teaches recording new rules for forms by querying a history database for suitable parameters (p. 7, par. 88). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claims 23 and 24**, while Weinberg does not explicitly teach fuzzy form detection, Sidles teaches a method of fuzzy form detection comparing a form being submitted to all forms in a history database (p. 9, par. 86-90), choosing a form from said session history which is most similar to said form being submitted. Sidles teaches generating logic rules for the form to be used to complete the form fields (p. 9, par. 90), compare to *recording said form merging instructions so that said*

*recorded form merging instructions applied to said form chosen from said session history result in a form identical to said form being submitted.* That is, the dictionary database is adjusted so that the automatic filler program and the rules engine will fill out that form the next time it is encountered (p. 10, par. 91, l. 1-8). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 25**, while Weinberg does not explicitly teach that the replay engine executes a test script with parser extensions and form merging instruction, Sidles teaches an automatic form filling system using fuzzy logic where a history unit generates a new set of rules based on form context (p. 9, par. 88, 90), i.e., *recording at least one context-full test script* for filling the form. Sidles also teaches that a replay engine is capable of executing the test script. Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would

have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding dependent claim 26**, claim 26 is directed to substantially similar subject matter as claimed in dependent claim 18, and is rejected along the same rationale.

**Regarding dependent claim 27,** Weinberg teaches a page-level API (Col. 19, l. 11-50; Fig. 8). Weinberg teaches an extensible document parser operable to determine at least one parser extension, where the parser extension includes a replay instruction specifying at least one parser addition and parameters for the parser addition, since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60), including the Action Tracker and Load Wizard containing replay instructions specifying parser additions; as well as parser extensions in the form of other external client applications (Col. 18, l. 40-59). Weinberg teaches that the replay engine of the test scenario recorder utilizes the document parsers.

**Regarding independent claim 28**, Weinberg teaches a method of comparing a form being submitted to one form in a session history, since Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33) using session history forms.

While Weinberg does not explicitly teach generating data based upon differences resulting from the comparing step, for each form in the session history, or recording

form merging instructions for fuzzy form detection, Sidles teaches a method of fuzzy form detection where a form is chosen from a database of forms which is similar to the form being submitted, and merged with past form data to produce a new filled form (p. 6, par. 61-62; p. 9, par. 86-90). Sidles teaches that the comparison and generating steps are performed for each form in the session history, since all forms from the previous sessions are stored in the database for comparison against the submitted form.

Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding independent claim 30**, Weinberg teaches an automated testing tool for testing and monitoring web applications (Col. 2, l. 30 – Col. 3, l. 37; Col. 8, l. 40-67); including a processor and memory (Col. 7, l. 51-Col. 8, l. 12; claim 13). Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33); compare to *said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user.*

Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37). Weinberg teaches that the recorder and replay engine have a context full API, since the test scenario is generated using information specific to the site context, i.e., server access log files and associated site usage information (Col. 33, l. 40-Col. 35, l. 20). Weinberg teaches a page-level API (Col. 19, l. 11-50; Fig. 8).

Weinberg teaches an extensible document parser operable to determine at least one parser extension, where the parser extension includes a replay instruction specifying at least one parser addition and parameters for the parser addition, since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60), including the Action Tracker and Load Wizard containing replay instructions specifying parser additions; as well as parser extensions in the form of other external client applications (Col. 18, l. 40-59). Weinberg teaches a library of parser additions, since Weinberg teaches the addition multiple plug-ins, each of which implements a specific parsing algorithm (Col. 17, l. 53-Col. 18, l. 20), including an algorithm for parsing hyperlinks, the Link Doctor. Weinberg teaches an algorithm for parsing HTML forms, the Dynamic Scan feature (Col. 25, l. 50-Col. 26, l. 51), including different configurations, i.e., extensions, of the Dynamic Scan feature including a standard web browser or a special integrated web browser (Col. 26, l. 41-51). Weinberg teaches a method of parsing hyperlinks by searching text between a left and right boundary string (Col. 20, l. 62-Col. 21, l. 60).

Weinberg teaches recording a page level test script comprising parser extensions, since Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37), and since Weinberg teaches a variety of plug-in modules for the document parser (Col. 17, l. 53-60). Weinberg does not explicitly teach form merging instructions including instructions for merging an HTML form and one context-full script form to produce a form to be submitted. However, Sidles teaches a reference to a previously downloaded form in a test script, i.e., to see if the user made changes to a field, and instructions for merging the form and scripts containing previously submitted form information to automatically fill out a form to be submitted (p. 12, par 120-123). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**Regarding independent claim 31**, Weinberg teaches an automated testing tool for testing and monitoring web applications (Col. 2, l. 30 – Col. 3, l. 37; Col. 8, l. 40-67); including a processor and memory (Col. 7, l. 51-Col. 8, l. 12; claim 13). Weinberg teaches that the testing tool has a Dynamic Scan feature for scanning dynamic HTML code, i.e., client side executable code, which may be set to automatically fill in dynamic

forms without interaction from a user (Col. 23, l. 15-Col. 26, l. 19, especially col. 24, l. 1-33); compare to *said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user.* Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37). Weinberg teaches that the recorder and replay engine have a context full API, since the test scenario is generated using information specific to the site context, i.e., server access log files and associated site usage information (Col. 33, l. 40-Col. 35, l. 20). Weinberg teaches a page-level API (Col. 19, l. 11-50; Fig. 8).

Weinberg teaches an algorithm for parsing HTML forms, the Dynamic Scan feature (Col. 25, l. 50-Col. 26, l. 51), including different configurations, i.e., extensions, of the Dynamic Scan feature including a standard web browser or a special integrated web browser (Col. 26, l. 41-51). Weinberg teaches a method of parsing hyperlinks by searching text between a left and right boundary string (Col. 20, l. 62-Col. 21, l. 60). Weinberg teaches recording a page level test script, since Weinberg teaches that the testing tool has a recorder operable to record a context full load testing script of a test scenario, and a replay engine to execute the test script (Col. 32, l. 23- Col. 33, l. 37).

Weinberg does not explicitly teach form merging instructions including instructions for merging an HTML form and one context-full script form to produce a form to be submitted. However, Sidles teaches a reference to a previously downloaded form in a test script, i.e., to see if the user made changes to a field, and instructions for merging the form and scripts containing previously submitted form information to

automatically fill out a form to be submitted (p. 12, par 120-123). Both Weinberg and Sidles are analogous art, being directed toward the testing and management of web applications. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply Sidles' form filling system with test scripts to Weinberg's automated testing tool with automatic filling of dynamic HTML forms, so that the testing tool of Weinberg would have the benefit of an self learning method of filling out forms which would improve performance on future occasions based on historic results (Sidles, p. 2, par. 17).

**(10) Response to Argument**

Beginning on page six of the Appeal Brief ("the Brief"), Appellants argue the following issues which are addressed below.

a. "**Grounds of Rejection No. 1**" (pages 6-9 of the Brief).

Applicants argue that the Weinberg patent does not disclose "client-side executable code" as claimed in independent claims 1 and 29, which recite "parsing said client side executable code" (p. 7-9 of the Brief). Applicant's define "client-side executable code" as software code that is executed within the web browser on the client computer, and give examples of "client-side executable code" including JavaScript code, Java applets, VB script, ActiveX controls, and other browser plug-ins.

Weinberg discloses a testing tool ("Astra") which parses "client-side executable code" as claimed in independent claims 1 and 29, because Weinberg teaches parsing the URLs of a website to derive content objects, including HTML documents, mail messages, **Java applets** and aglets, audio files, video files, and **applications** (col. 8, l. 34-52, especially l. 47-52, emphasis added).

Weinberg discloses that during the web site scanning process, the Astra testing tool communicates with one or more web server applications, which may run on a single computer, or on multiple computers at the location of the client computer (col. 18, l. 60-col. 19, l. 3; Fig. 7). Weinberg discloses automatically accessing and filling in dynamic HTML forms from the client side, i.e., the client browser (col. 23, l. 15-col. 26, l. 19). Weinberg discloses that the Astra testing tool resides on the client computer, and therefore all code of a web site is accessed and executed through the client computer, and recorded into a test script by the Astra testing tool.

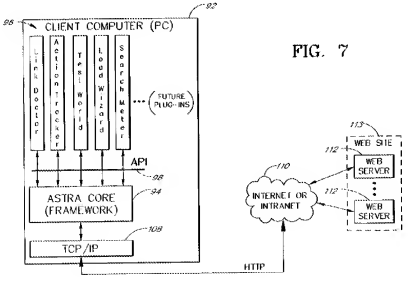


FIG. 7

Therefore, Weinberg discloses a testing tool in which the code of the web site to be tested is accessed and executed from the client side, and also discloses parsing client side executable code, including Java applets and applications, as well as client-side HTML forms, using the Astra testing tool, in order to record a test script.

While appellants argue that Weinberg's dynamic scan feature is directed to the scanning and mapping of web pages that are dynamically generated on the server (i.e., by a web site), appellants' arguments do not address the other disclosed features of Weinberg which scan and record the content of a web site, including the Java applets and applications, which appellants admit can be executed on the client side (col. 8, l. 34-52).

However, Weinberg's dynamic scan feature also discloses executing an HTML form from the client browser, and submitting the form to a web server for further processing, thereby disclosing code that is executable both on the client and server. Figure 12 shows that the HTML form page is executed from the client computer's web browser, item 170, and sent to the web server. While Appellants argue that an HTML form is not "client-side executable code", Weinberg discloses that the client computer initiates the transaction with the server by submitting the form and data set, and therefore the form and submission request constitutes code executed by the client that is sent to the server. Figure 12, and col. 24, l. 33-col. 25, l. 50 of Weinberg disclose a process of exchanging executable software code between a client and server computer, in which software code is executed both on the client and the server.

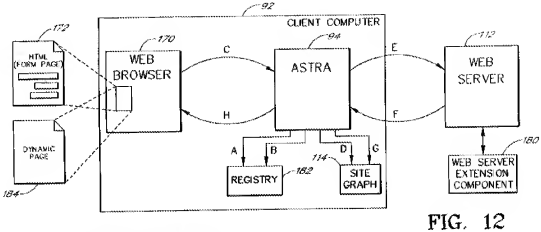


FIG. 12

A ASTRA MODIFIES BROWSER CONFIGURATION IN REGISTRY TO SET ASTRA AS PROXY
B. ASTRA LAUNCHES BROWSER, AND THEN RESTORES ORIGINAL BROWSER CONFIGURATION WITHIN REGISTRY
C BROWSER PASSES HTTP MESSAGE TO ASTRA IN RESPONSE TO SUBMISSION OF FORM
D ASTRA EXTRACTS DATA SET AND STORES IN SITE GRAPH FOR FUTURE USE
E. ASTRA FORWARDS HTTP MESSAGE TO WEB SERVER
F. WEB SERVER RETURNS DYNAMICALLY-GENERATED WEB PAGE
G WEB SERVER PARSES DYNAMICALLY-GENERATED PAGE AND UPDATES SITE GRAPH
H ASTRA FORWARDS PAGE TO BROWSER TO CREATE IMPRESSION OF REGULAR BROWSING

Figure 12 of Weinberg (Step C-D disclose client-side executable code and steps E-H disclose the limitation of claim 1, "parsing said client side executable code so as to determine a subsequent state of the network application free of interaction with a user...").

While appellant's arguments against Weinberg hinge on the assumption that Weinberg does not disclose "client-side executable code", it is noted that claim 1 recites a "network application", thereby claiming both a client and a network. It is respectfully noted that while independent claims 1 and 29 recite "parsing said client side executable code so as to determine a subsequent state of the network application free of

interaction with a user," (Claim 1), the claimed invention does not specify whether the client side executable code is parsed on the client or the server. As discussed above, Weinberg discloses parsing client side executable code, as well as executing code from the client side and parsing code generated by both the client and server (Fig. 12; col. 24, l. 33-col. 25, l. 50).

Weinberg specifically discloses that the testing tool parses the same types of client-side executable code which have been listed as examples in appellants' arguments (p. 7, par. 4 of the Brief), including HTML documents, mail messages, Java applets and aglets, audio files, video files, and applications (col. 8, l. 34-52).

b. "**Grounds of Rejection No. 2**" (pages 9-10 of the Brief).

Appellants present arguments in regard to the Sidles publication, which was relied upon in combination with the Weinberg patent for the rejections of dependent claims 13-27 and independent claims 28, 30 and 31, as unpatentable under 35 U.S.C. 103(a).

Regarding claims 13-27, 30, and 31, appellants assert that Sidles does not teach the claim limitations as set forth in the claim rejections (in combination with Weinberg), but present no substantive arguments to uphold their assertion that the claim rejections do not make a prima facie case of obviousness (p. 9, par. 4 of the Brief). It is respectfully noted that for claims 13-27, 30, and 31, appellants do not list any of the

claim limitations which are alleged not to be disclosed by Sidles, nor do they present arguments in regard to the motivation to combine the references.

As set forth in the claim rejections for claims 13-27, 30, and 31, Sidles does teach, in combination with Weinberg, each and every limitation of the claims, and for the reasons of record, it would have been obvious to combine Weinberg and Sidles.

Regarding independent claim 28, appellants argue that Sidles does not teach the claim limitations: "*...comparing a form to be submitted to at least one form in a session history; generating data based on differences resulting from the comparing step;...*"

Weinberg is relied upon to teach the limitation "*...comparing a form to be submitted to at least one form in a session history;*" because Weinberg teaches recording a session history for automatically filling in HTML forms (col. 24, l. 1-33; col. 23, l. 15-col. 26, l. 19). Sidles teaches the limitation, " *generating data based on differences resulting from the comparing step;*" because Sidles teaches comparing two or more copies of a form in a history database, and generating a new set of rules data for the form (p. 9, par. 0087-0088). While appellants argue that Sidles does not generate data based upon the differences of any comparison, but that "Sidles merely attempts to apply rules" (p. 10, par. 3 of the Brief), the limitation of claim 28 recites *generating data*, and Sidles teaches *rules data*, therefore Sidles does teach the limitation " *generating data based on differences resulting from the comparing step*".

Appellant's remaining arguments against Sidles assert that Sidles does not teach the limitations of claim 28, *...choosing one of the forms in said session history having*

*the greatest similarity to said form to be submitted based upon the generating step results; and applying form merging instructions to said session history form to obtain a resulting form that is substantially identical to said form to be submitted.*" Appellants submit that "because Sidles lacks the generating step" Sidles also lacks the remaining claim limitations (p. 10, par. 4 of the Brief). However, Sidles does teach the generating step, and the above claim limitations, because Sides teaches storing past forms, determining which forms are similar, choosing a similar form, and applying form merging instructions using fuzzy logic (p. 6, par. 0061-0062; p. 9, par. 0086-0090). Also see par. 0018 of Sidles which provides an overview of the invention. It is believed that Sidles, in combination with Weinberg, which also discloses automated form completion, discloses each and every limitation of claim 28.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Amelia Rutledge/

Examiner, Art Unit 2176

Conferees:

*/Doug Hutton/*
Doug Hutton
Supervisory Primary Examiner
Technology Center 2100

/Rachna S Desai/
Primary Examiner, Art Unit 2176